L3/M1, 2005–2006

Algorithmique effective – TD 2

1 Hors-d'œuvre

Exercice 1
Glouton.
Écrire en C++ un algorithme glouton pour le problème du sac-à-dos. La première ligne de l'entrée est constituée d'un entier qui est le poids total supporté par le sac-à-dos. La deuxième ligne contient le nombre d'objets qui vont suivre. Sur chacune des lignes suivantes un objet est représenté, par deux entiers : d'abord son poids, puis sa valeur.

La sortie est un entier représentant la valeur qu'on peut emporter, donnée par l'algorithme glouton.

2 Plat chaud

Exercice 2
Crypt Kicker. PC : 110204 ; UVa : 843.
A common but insecure method of encrypting text is to permute the letters of the alphabet. In other words, each letter of the alphabet is consistently replaced in the text by some other letter. To ensure that the encryption is reversible, no two letters are replaced by the same letter.

Your task is to decrypt several encoded lines of text, assuming that each line uses a different set of replacements, and that all words in the decrypted text are from a dictionary of known words.

Input
The input consists of a line containing an integer n, followed by n lowercase words, one per line, in alphabetical order. These n words compose the dictionary of words which may appear in the decrypted text. Following the dictionary are several lines of input. Each line is encrypted as described above.

There are no more than 1,000 words in the dictionary. No word exceeds 16 letters. The encrypted lines contain only lower case letters and spaces and do not exceed 80 characters in length.

Output
Decrypt each line and print it to standard output. If there are multiple solutions, any one will do. If there is no solution, replace every letter of the alphabet by an asterisk.

Sample Input

```
6
and
dick
jane
puff
spot
yertle
bjvg xsb hxsn xsb qymm xsb rqat xsb pnetfn
xxxx yyy zzzz www yyyy aaa bbbb ccc dddddd
```

Sample Output

```
dick and jane and puff and spot and yertle
**** *** **** *** **** *** **** *** ******
```

(Pour prendre en compte la répétition des lettres, on pourra associer à chaque lettre du mot la position de sa première occurence. Par exemple 1211321 pour `abaacba`*. On peut coder cela dans une paire d'*`unsigned long`*.)*

3 Dessert

Voici un exposé du problème de l'affectation, à résoudre par branch-and-bound.

This is a basic optimization problem that nevertheless already has many interesting aspects, and even considerable practicle importance. In the assignment problem, there are n agents and n tasks. In principle each agent can perform any of the tasks, but at different costs. The task is to allocate one task to every agent, so that the sum of the costs is minimized. "Assignment" problem is another name for the "bipartite weighted matching" problem.

Mathematically the problem can be formulated as follows. Given an $n \times n$ costmatrix M, we should determine a permutation π of n numbers, so that $C(\pi)$, the cost with respect to π, given by $C(\pi) = \sum_i M(i, pi(i))$ is minimized. $\pi(i)$ can be interpreted as the task performed by agent i.

Upper bounds can be obtained by applying the greedy algorithm, assigning the lightest task to the agents in order, or assigning the cheapest agent to each task in order. Lower bounds are obtained as the maximum of the sum of the minima in the rows or the sum of the minima in the columns. Once a certain number of choices are made, the upper bounds and lower bounds are of course determined by only looking at the remaining submatrix and adding the already incured costs. For the assignment we start with the first agent and consider the choices in order of increasing cost. The search is performed in DFS (depth first) order.

As an example we consider an assignment problem with $n = 4$ and the following cost matrix :

```
      1   2   3   4
   ------------------
a | 11  12  18  40 |
b | 14  15  13  22 |
c | 11  17  19  23 |
d | 17  14  20  28 |
   ------------------
```

Écrire un algorithme branch-and-bound pour ce problème. Dans le rapport, on pourra modifier les algorithmes pour les bornes sup et inf afin de visualiser leur impact.

3.1 The great wall game

Écrire un programme pour le problème de la finale 2005 au verso. On pourra faire le lien avec le problème de l'affectation ci-dessus.